Microsoft Knowledge Base Article - 257819

HOWTO: Use ADO with Excel Data from Visual Basic or VBA

Applies To

This article was previously published under Q257819

SUMMARY

This article discusses the use of ActiveX Data Objects (ADO) with Microsoft Excel spreadsheets as a data source. The article also highlights syntax issues and limitations specific to Excel. This article does not discuss OLAP or PivotTable technologies or other specialized uses of Excel data.

For additional information, click the article number below to view the article in the Microsoft Knowledge Base:

303814 HOWTO: Use ADOX with Excel Data from Visual Basic or VBA

MORE INFORMATION

INTRODUCTION

The rows and columns of a Microsoft Excel spreadsheet closely resemble the rows and columns of a database table. As long as users keep in mind that Microsoft Excel is not a relational database management system, and recognize the limitations that this fact imposes, it often makes sense to take advantage of Excel and its tools to store and analyze data.

Microsoft ActiveX Data Objects makes it possible to treat an Excel workbook as if it were a database. This article discusses how to accomplish this in the following sections:

- Connect to Excel with ADO
- Retrieve and Edit Excel Data with ADO
- Retrieve Data Source Structure (Metadata) from Excel

NOTE: The testing for this article was conducted with Microsoft Data Access Components (MDAC) 2.5 on Microsoft Windows 2000 with Visual Basic 6.0 Service Pack 3 and Excel 2000. This article may not acknowledge or discuss differences in behavior that users may observe with different versions of MDAC, Microsoft Windows, Visual Basic, or Excel.

Connect to Excel with ADO

ADO can connect to an Excel data file with either one of two OLE DB Providers included in MDAC:

- Microsoft Jet OLE DB Provider -or-
- Microsoft OLE DB Provider for ODBC Drivers

How to Use the Microsoft Jet OLE DB Provider

The Jet Provider requires only two pieces of information in order to connect to an Excel data source: the path, including the file name, and the Excel file version.

Jet Provider Using a Connection String

```
Dim cn as ADODB.Connection
Set cn = New ADODB.Connection
With cn
          .Provider = "Microsoft.Jet.OLEDB.4.0"
          .ConnectionString = "Data Source=C:\MyFolder\MyWorkbook.xls;" & _
"Extended Properties=Excel 8.0;"
          .Open
```

```
End With
```

Provider Version: It is necessary to use the Jet 4.0 Provider; the Jet 3.51 Provider does not support the Jet ISAM drivers. If you specify the Jet 3.51 Provider, at run time you receive the following error message:

> Couldn't find installable ISAM.

Excel Version: Specify Excel 5.0 for an Excel 95 workbook (version 7.0 of Excel), and Excel 8.0 for an Excel 97, Excel 2000, or Excel 2002 (XP) workbook (versions 8.0, 9.0, and 10.0 of Excel).

Jet Provider Using the Data Link Properties Dialog Box

If you use the ADO Data Control or the Data Environment in your application, then the **Data Link Properties** dialog box is displayed to gather the necessary connection settings.

1. On the **Provider** tab, select the Jet 4.0 Provider; the Jet 3.51 Provider does not support the Jet ISAM drivers. If you specify the Jet 3.51 Provider, at run time you receive the following error message:

   > Couldn't find installable ISAM.

2. On the **Connection** tab, browse to your workbook file. Ignore the "User ID" and "Password" entries, because these do not apply to an Excel connection. (You cannot open a password-protected Excel file as a data source. There is more information on this topic later in this article.)
3. On the **All** tab, select **Extended Properties** in the list, and then click **Edit Value**. Enter **Excel 8.0;** separating it from any other existing entries with a semicolon (;). If you omit this step, you receive an error message when you test your connection, because the Jet Provider expects a Microsoft Access database unless you specify otherwise.
4. Return to the **Connection** tab and click **Test Connection**. Note that a message box appears informing you that the process has succeeded.

Other Jet Provider Connection Settings

Column headings: By default, it is assumed that the first row of your Excel data source contains columns headings that can be used as field names. If this is not the case, you must turn this setting off, or your first row of data "disappears" to be used as field names. This is done by adding the optional **HDR=** setting to the **Extended Properties** of the connection string. The default, which does not need to be specified, is **HDR=Yes**. If you do not have column headings, you need to specify **HDR=No**; the provider names your fields F1, F2, etc. Because the **Extended Properties** string now contains multiple values, it must be enclosed in double quotes itself, plus an additional pair of double quotes to tell Visual Basic to treat the first set of quotes as literal values, as in the following example (where extra spaces have been added for visual clarity).

```
.ConnectionString = "Data Source=C:\MyFolder\MyWorkbook.xls;" & _
"Extended Properties=" " Excel 8.0; HDR=No;" " "
```

Using Microsoft OLE DB Provider for ODBC Drivers

The provider for ODBC drivers (which this article refers to as the "ODBC Provider" for the sake of brevity) also requires only two (2) pieces of information in order to connect to an Excel data source: the driver name, and the workbook path and filename.

IMPORTANT: An ODBC connection to Excel is read-only by default. Your ADO Recordset **LockType** property setting does not override this connection-level setting. You must set **ReadOnly** to **False** in your connection string or your DSN configuration if you want to edit your data. Otherwise, you receive the following error message:

> Operation must use an updateable query.

ODBC Provider Using a DSN-Less Connection String

```
Dim cn as ADODB.Connection
Set cn = New ADODB.Connection
With cn
        .Provider = "MSDASQL"
        .ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
```

```
"DBQ=C:\MyFolder\MyWorkbook.xls; ReadOnly=False;"
          .Open
End With
```

ODBC Provider Using a Connection String with a DSN

```
Dim cn as ADODB.Connection
Set cn = New ADODB.Connection
With cn
          .Provider = "MSDASQL"
          .ConnectionString = "DSN=MyExcelDSN;"
          .Open
End With
```

ODBC Provider Using the Data Link Properties Dialog Box

If you use the ADO Data Control or the Data Environment in your application, then the **Data Link Properties** dialog box is displayed to gather the necessary connection settings.

1. On the **Provider** tab, select **Microsoft OLE DB Provider for ODBC Drivers**.
2. On the **Connection** tab, select the existing DSN that you want to use, or choose **Use connection string**. This brings up the standard DSN configuration dialog box to gather the necessary connection settings. Remember to deselect the default read-only setting if desired, as mentioned previously.
3. Return to the **Connection** tab, and click **Test Connection**. Note that a message box appears informing you that the process has succeeded.

Other ODBC Provider Connection Settings

Column headings: By default, it is assumed that the first row of your Excel data source contains columns headings, which can be used as field names. If this is not the case, you must turn this setting off, or your first row of data "disappears" to be used as field names. This is done by adding the optional **FirstRowHasNames=** setting to the connection string. The default, which does not need to be specified, is **FirstRowHasNames=1**, where **1 = True**. If you do not have column headings, you need to specify **FirstRowHasNames=0**, where **0 = False**; the driver names your fields F1, F2, and so forth. This option is not available in the DSN configuration dialog box.

However, due to a bug in the ODBC driver, specifying the **FirstRowHasNames** setting currently has no effect. In other words, the Excel ODBC driver (MDAC 2.1 and later) always treats the first row in the specified data source as field names. For additional informationon the Column Heading bug, click the article number below to view the article in the Microsoft Knowledge Base:

288343 BUG: Excel ODBC Driver Disregards the FirstRowHasNames or Header Setting

Rows to Scan: Excel does not provide ADO with detailed schema information about the data it contains, as a relational database would. Therefore, the driver must scan through at least a few rows of the existing data in order to make an educated guess at the data type of each column. The default for "Rows to Scan" is eight (8) rows. You can specify an integer value from one (1) to sixteen (16) rows, or you can specify zero (0) to scan all existing rows. This is done by adding the optional **MaxScanRows=** setting to the connection string, or by changing the **Rows to Scan** setting in the DSN configuration dialog box.

However, due to a bug in the ODBC driver, specifying the Rows to Scan (MaxScanRows) setting currently has no effect. In other words, the Excel ODBC driver (MDAC 2.1 and later) always scans the first 8 rows in the specified data source in order to determine each column's datatype.

For additional information about the Rows to Scan bug, including a simple workaround, click the article number below to view the article in the Microsoft Knowledge Base:

189897 XL97: Data Truncated to 255 Characters with Excel ODBC Driver

Other Settings: If you construct your connection string by using the **Data Link Properties** dialog box, you may notice some other **Extended Properties** settings added to the connection string that are not absolutely necessary, such as:

```
... DefaultDir=C:\WorkbookPath;DriverId=790;FIL=excel 8.0;MaxBufferSize=2048;PageTimeout=5;
```

"Collating Sequence" Error Message in the Visual Basic Editor

In the Visual Basic design environment with certain versions of MDAC, you may see the following error message the first time your program connects to an Excel data source at design time:

> Selected collating sequence not supported by the operating system.

This message appears only in the IDE and will not appear in the compiled version of the program. For additional information, click the article number below to view the article in the Microsoft Knowledge Base:

> 246167 PRB: Collating Sequence Error Opening ADODB Recordset the First Time Against an Excel XLS

Considerations That Apply to Both OLE DB Providers

A Caution about Mixed Data Types

As stated previously, ADO must guess at the data type for each column in your Excel worksheet or range. (This is not affected by Excel cell formatting settings.) A serious problem can arise if you have numeric values mixed with text values in the same column. Both the Jet and the ODBC Provider return the data of the majority type, but return NULL (empty) values for the minority data type. If the two types are equally mixed in the column, the provider chooses numeric over text.

For example:

- In your eight (8) scanned rows, if the column contains five (5) numeric values and three (3) text values, the provider returns five (5) numbers and three (3) null values.
- In your eight (8) scanned rows, if the column contains three (3) numeric values and five (5) text values, the provider returns three (3) null values and five (5) text values.
- In your eight (8) scanned rows, if the column contains four (4) numeric values and four (4) text values, the provider returns four (4) numbers and four (4) null values.

As a result, if your column contains mixed values, your only recourse is to store numeric values in that column as text, and to convert them back to numbers when needed in the client application by using the Visual Basic **VAL** function or an equivalent.

To work around this problem for read-only data, enable **Import Mode** by using the setting "IMEX=1" in the Extended Properties section of the connection string. This enforces the **ImportMixedTypes=Text** registry setting. However, note that updates may give unexpected results in this mode. For additional information about this setting, click the article number below to view the article in the Microsoft Knowledge Base:

> 194124 PRB: Excel Values Returned as NULL Using DAO OpenRecordset

You Cannot Open a Password-Protected Workbook

If the Excel workbook is protected by a password, you cannot open it for data access, even by supplying the correct password with your connection settings, unless the workbook file is already open in the Microsoft Excel application. If you try, you receive the following error message:

> Could not decrypt file.

For additional information, click the article number below to view the article in the Microsoft Knowledge Base:

> 211378 XL2000: "Could Not Decrypt File" Error with Password Protected File

Retrieve and Edit Excel Data with ADO

This section discusses two aspects of working with your Excel data:

- How to select data -and-
- How to change data

How to Select Data

There are several ways to select data. You can:

- Select Excel data with code.
- Select Excel data with the ADO Data control.
- Select Excel data with Data Environment commands.

Select Excel Data with Code

Your Excel data may be contained in your workbook in one of the following:

- An entire worksheet.
- A named range of cells on a worksheet.
- An unnamed range of cells on a worksheet.

Specify a Worksheet

To specify a worksheet as your recordsource, use the worksheet name followed by a dollar sign and surrounded by square brackets. For example:

```
strQuery = "SELECT * FROM [Sheet1$]"
```

You can also delimit the worksheet name with the slanted single quote character (`) found on the keyboard under the tilde (~). For example:

```
strQuery = "SELECT * FROM `Sheet1$`"
```

Microsoft prefers the square brackets, which are the standing convention for problematic database object names.

If you omit both the dollar sign and the brackets, or just the dollar sign, you receive the following error message:

... the Jet database engine could not find the specified object

If you use the dollar sign but omit the brackets, you will see the following error message:

Syntax error in FROM clause.

If you try to use ordinary single quotes, you receive the following error message:

Syntax error in query. Incomplete query clause.

Specify a Named Range

To specify a named range of cells as your recordsource, simply use the defined name. For example:

```
strQuery = "SELECT * FROM MyRange"
```

Specify an Unnamed Range

To specify an unnamed range of cells as your recordsource, append standard Excel row/column notation to the end of the sheet name in the square brackets. For example:

```
strQuery = "SELECT * FROM [Sheet1$A1:B10]"
```

A caution about specifying worksheets: The provider assumes that your table of data begins with the upper-most, left-most, non-blank cell on the specified worksheet. In other words, your table of data can begin in Row 3, Column C without a problem. However, you cannot, for example, type a worksheeet title above and to the left of the data in cell A1.

A caution about specifying ranges: When you specify a worksheet as your recordsource, the provider adds new records below existing records in the worksheet as space allows. When you specify a range (named or unnamed), Jet also adds new records below the existing records in the range as space allows. However, if you requery on the original range, the resulting recordset does not include the newly added records outside the range.

With MDAC versions prior to 2.5, when you specify a named range, you cannot add new records beyond the defined limits of the range, or you receive the following error message:

> Cannot expand named range.

Select Excel Data with the ADO Data Control

After you specify the connection settings for your Excel data source on the **General** tab of the ADODC **Properties** dialog box, click on the **Recordsource** tab. If you choose a CommandType of adCmdText, you can enter a SELECT query in the **Command Text** dialog box with the syntax described previously. If you choose a CommandType of adCmdTable, and you are using the Jet Provider, the drop-down list displays both the named ranges and worksheet names that are available in the selected workbook, with named ranges listed first.

This dialog box properly appends the dollar sign to worksheet names, but does not add the necessary square brackets. As a result, if you simply select a worksheet name and click **OK**, you receive the following error message later:

> Syntax error in FROM clause.

You must manually add the square brackets around the worksheet name. (This combo box does allow editing.) If you are using the ODBC Provider, you see only named ranges listed in this drop-down list. However, you can manually enter a worksheet name with the appropriate delimiters.

Select Excel Data with Data Environment Commands

After setting up the Data Environment Connection for your Excel data source, create a new **Command** object. If you choose a **Source of Data** of **SQL Statement**, you can enter a query in the textbox using the syntax described previously. If you choose a **Source of Data** of **Database Object**, select **Table** in the first drop-down list, and you are using the Jet Provider, the drop-down list displays both named ranges and worksheet names available in the selected workbook, with named ranges listed first. (If you choose a worksheet name in this location, you do not need to add square brackets around the worksheet name manually as you do for the ADO Data Control.) If you are using the ODBC Provider, you see only named ranges listed in this drop-down list. However, you can manually enter a worksheet name.

How to Change Excel Data: Edit, Add, and Delete

Edit

You can edit Excel data with the normal ADO methods. Recordset fields which correspond to cells in the Excel worksheet containing Excel formulas (beginning with "=") are read-only and cannot be edited. Remember that an ODBC connection to Excel is read-only by default, unless you specify otherwise in your connection settings. See earlier under "Using the Microsoft OLE DB Provider for ODBC Drivers."

Add

You can add records to your Excel recordsource as space allows. However, if you add new records outside the range that you originally specified, these records are not visible if you requery on the original range specification. See earlier under "A caution about specifying ranges."

In certain circumstances, when you use the **AddNew** and **Update** methods of the ADO **Recordset** object to insert new rows of data into an Excel table, ADO may insert the data values into the wrong columns in Excel. For additional information, click the article number below to view the article in the Microsoft Knowledge Base:

> 314763 FIX: ADO Inserts Data into Wrong Columns in Excel

Delete

You are more restricted in deleting Excel data than data from a relational data source. In a relational database, "row" has no meaning or existence apart from "record"; in an Excel worksheet, this is not true. You can delete values in fields (cells). However, you cannot:

1. Delete an entire record at once or you receive the following error message:

   ```
   Deleting data in a linked table is not supported by this ISAM.
   ```

   You can only delete a record by blanking out the contents of each individual field.
2. Delete the value in a cell containing an Excel formula or you receive the following error message:

   ```
   Operation is not allowed in this context.
   ```

3. You cannot delete the empty spreadsheet row(s) in which the deleted data was located, and your recordset will continue to display empty records corresponding to these empty rows.

A caution about editing Excel data with ADO: When you insert text data into Excel with ADO, the text value is preceded with a single quote. This may cause problems later in working with the new data.

Retrieve Data Source Structure (Metadata) from Excel

You can retrieve data about the structure of your Excel data source (tables and fields) with ADO. Results differ slightly between the two OLE DB Providers, although both return at least the same small number of useful fields of information. This metadata can be retrieved with the **OpenSchema** method of the ADO **Connection** object, which returns an ADO **Recordset** object. You can also use the more powerful Microsoft ActiveX Data Objects Extensions for Data Definition Language and Security (ADOX) library for this purpose. In the case of an Excel data source however, where a "table" is either a worksheet or a named range, and a "field" is one of a limited number of generic datatypes, this additional power is not useful.

Query Table Information

Of the various objects available in a relational database (tables, views, stored procedures, and so forth), an Excel data source exposes only table equivalents, consisting of the worksheets and the named ranges defined in the specified workbook. Named ranges are treated as "Tables" and worksheets are treated as "System Tables," and there is not much useful table information you can retrieve beyond this "table_type" property. You request a list of the available tables in the workbook with the following code:

```
Set rs = cn.OpenSchema(adSchemaTables)
```

The Jet Provider returns a recordset with nine (9) fields, of which it populates only four (4):

- table_name
- table_type ("Table" or "System Table")
- date_created
- date_modified

The two date fields for a given table always show the same value, which appears to be the "date last modified." In other words, "date_created" is not reliable.

The ODBC Provider also returns a recordset with nine (9) fields, of which it populates only three (3):

- table_catalog, the folder in which the workbook is located.
- table_name.
- table_type, as noted earlier.

According to the ADO documentation, it is possible to retrieve a list of worksheets only, for example, by specifying the following additional criteria to the **OpenSchema** method:

```
Set rs = cn.OpenSchema(adSchemaTables, Array(Empty, Empty, Empty, "System Table"))
```

Unfortunately, this does not work against an Excel data source with MDAC versions later than 2.0, using either provider.

Query Field Information

Every field (column) in an Excel data source is one of the following datatypes:

- numeric (ADO datatype 5, adDouble)
- currency (ADO datatype 6, adCurrency)
- logical or boolean (ADO datatype 11, adBoolean)
- date (ADO datatype 7, adDate, using Jet; 135, adDBTimestamp, using ODBC)
- text (an ADO ad...Char type, such as 202, adVarChar, 200, adVarWChar or similar)

The numeric_precision for a numeric column is always returned as 15 (which is the maximum precision in Excel); the character_maximum_length of a text column is always returned as 255 (which is the maximum display width, but not the maximum length, of text in an Excel column). There is not much useful field information that you can obtain beyond the **data_type** property. You request a list of the available fields in a table with the following code:

```
Set rs = cn.OpenSchema(adSchemaTables, Array(Empty, Empty, "TableName", Empty))
```

The Jet Provider returns a recordset that contains 28 fields, of which it populates eight (8) for numeric fields and nine (9) for text fields. The useful fields are likely to be these:

- table_name
- column_name
- ordinal_position
- data_type

The ODBC Provider returns a recordset containing 29 fields, of which it populates ten (10) for numeric fields and 11 for text fields. The useful fields are the same as earlier.

Enumerate Tables and Fields and Their Properties

Visual Basic code (such as the following sample) can be used to enumerate the tables and columns in an Excel data source and the available fields of information about each. This sample outputs its results to a Listbox, List1, on the same form.

```
Dim cn As ADODB.Connection
Dim rsT As ADODB.Recordset
Dim intTblCnt As Integer, intTblFlds As Integer
Dim strTbl As String
Dim rsC As ADODB.Recordset
Dim intColCnt As Integer, intColFlds As Integer
Dim strCol As String
Dim t As Integer, c As Integer, f As Integer
Set cn = New ADODB.Connection
With cn
          .Provider = "Microsoft.Jet.OLEDB.4.0"
          .ConnectionString = "Data Source=" & App.Path & _
"\ExcelSrc.xls;Extended Properties=Excel 8.0;"
          '.Provider = "MSDASQL"
          '.ConnectionString = "Driver={Microsoft Excel Driver (*.xls)};" & _
"DBQ=" & App.Path & "\ExcelSrc.xls; "
          .CursorLocation = adUseClient
          .Open
End With
Set rsT = cn.OpenSchema(adSchemaTables)
intTblCnt = rsT.RecordCount
intTblFlds = rsT.Fields.Count
List1.AddItem "Tables:       " & intTblCnt
List1.AddItem "--------------------"
For t = 1 To intTblCnt
          strTbl = rsT.Fields("TABLE_NAME").Value
          List1.AddItem vbTab & "Table #" & t & ":        " & strTbl
          List1.AddItem vbTab & "--------------------"
          For f = 0 To intTblFlds - 1
                    List1.AddItem vbTab & rsT.Fields(f).Name & _
vbTab & rsT.Fields(f).Value
```

```
            Next
            List1.AddItem "--------------------"
            Set rsC = cn.OpenSchema(adSchemaColumns, Array(Empty, Empty, strTbl, Empty))
            intColCnt = rsC.RecordCount
            intColFlds = rsC.Fields.Count
            For c = 1 To intColCnt
                     strCol = rsC.Fields("COLUMN_NAME").Value
                     List1.AddItem vbTab & vbTab & "Column #" & c & ": " & strCol
                     List1.AddItem vbTab & vbTab & "--------------------"
                     For f = 0 To intColFlds - 1
                              List1.AddItem vbTab & vbTab & rsC.Fields(f).Name & _
    vbTab & rsC.Fields(f).Value
                     Next
                     List1.AddItem vbTab & vbTab & "--------------------"
                     rsC.MoveNext
                     Next
                     rsC.Close
                     List1.AddItem "--------------------"
                     rsT.MoveNext
    Next
    rsT.Close
    cn.Close
```

Use the Data View Window

If you create a data link to an Excel data source in the Visual Basic Data View window, the Data View window displays the same information that you can retrieve programmatically as described earlier. In particular, note that the Jet Provider lists both worksheets and named ranges under "Tables," where the ODBC Provider shows only named ranges. If you are using the ODBC Provider and have not defined any named ranges, the "Tables" list will be empty.

Excel Limitations

The use of Excel as a data source is bound by the internal limitations of Excel workbooks and worksheets. These include, but are not limited to:

- Worksheet size: 65,536 rows by 256 columns
- Cell contents (text): 32,767 characters
- Sheets in a workbook: limited by available memory
- Names in a workbook: limited by available memory

REFERENCES

For additional information about how to use ADO.NET to retrieve and modify records in an Excel workbook with Visual Basic .NET, click the following article number to view the article in the Microsoft Knowledge Base:

316934 HOW TO: Use ADO.NET to Retrieve and Modify Records in an Excel Workbook With Visual Basic .NET

For additional information, click the article numbers below to view the articles in the Microsoft Knowledge Base:

295646 HOWTO: Transfer Data from ADO Data Source to Excel with ADO

246335 HOWTO: Transfer Data from ADO Recordset to Excel with Automation

247412 INFO: Methods for Transferring Data to Excel from Visual Basic

278973 SAMPLE: ExcelADO Demonstrates How to Use ADO to Read and Write Data in Excel Workbooks

318373 HOW TO: Retrieve Metadata from Excel by Using the GetOleDbSchemaTable Method in Visual Basic .NET

For more information, refer to the following Microsoft Training & Certification course:

Microsoft Corporation 1301 Mastering Office 2000 Solution Development

The information in this article applies to:

- Microsoft Excel 2000
- Microsoft Visual Basic Learning Edition for Windows 6.0
- Microsoft Visual Basic Professional Edition for Windows 6.0
- Microsoft Visual Basic Enterprise Edition for Windows 6.0
- Microsoft Visual Basic Enterprise Edition for Windows 6.0 SP3
- Microsoft Visual Basic for Applications 6.0
- ActiveX Data Objects (ADO) 2.1
- ActiveX Data Objects (ADO) 2.1 SP1
- ActiveX Data Objects (ADO) 2.1 SP2
- ActiveX Data Objects (ADO) 2.5
- Microsoft Data Access Components 2.1
- Microsoft Data Access Components 2.1 SP1
- Microsoft Data Access Components 2.1 SP2
- Microsoft Data Access Components 2.5
- Microsoft Excel 2002
- Microsoft Excel 97 for Windows
- Microsoft Excel for Windows 95

Last Reviewed: 5/8/2003 (3.0)

Keywords: kbhowto kbIISAM KB257819

Send **Print** **Help**